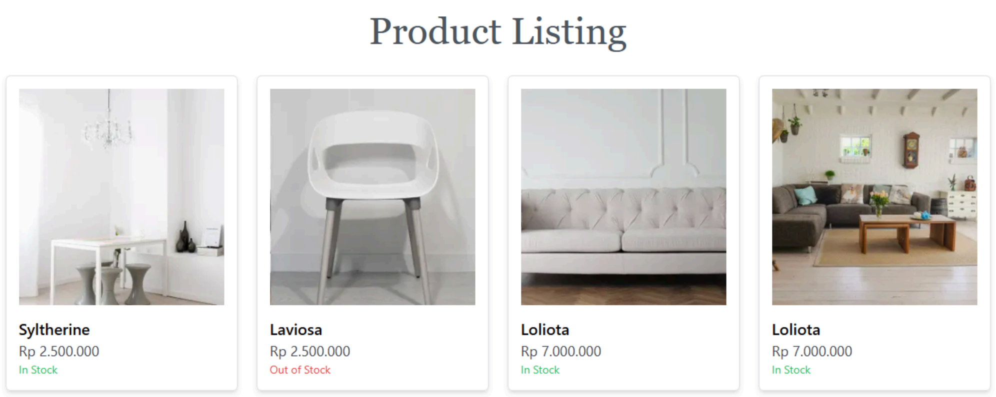
Day 4 - Building Dynamic Frontend Components

Focus on designing and developing dynamic frontend components to display marketplace data fetched from Sanity CMS or APIs. Emphasize modular, reusable component design, responsive design, and professional workflows.

Key Components to Build

1) Product Listing Component:

We first render all the data in grid layout to make a unique change in ui and give you a attractive and responsive ui we include some fields
Productname
Price
Image
Tags


Product Listing

Sylytherine
Rp 2.500.000
In Stock

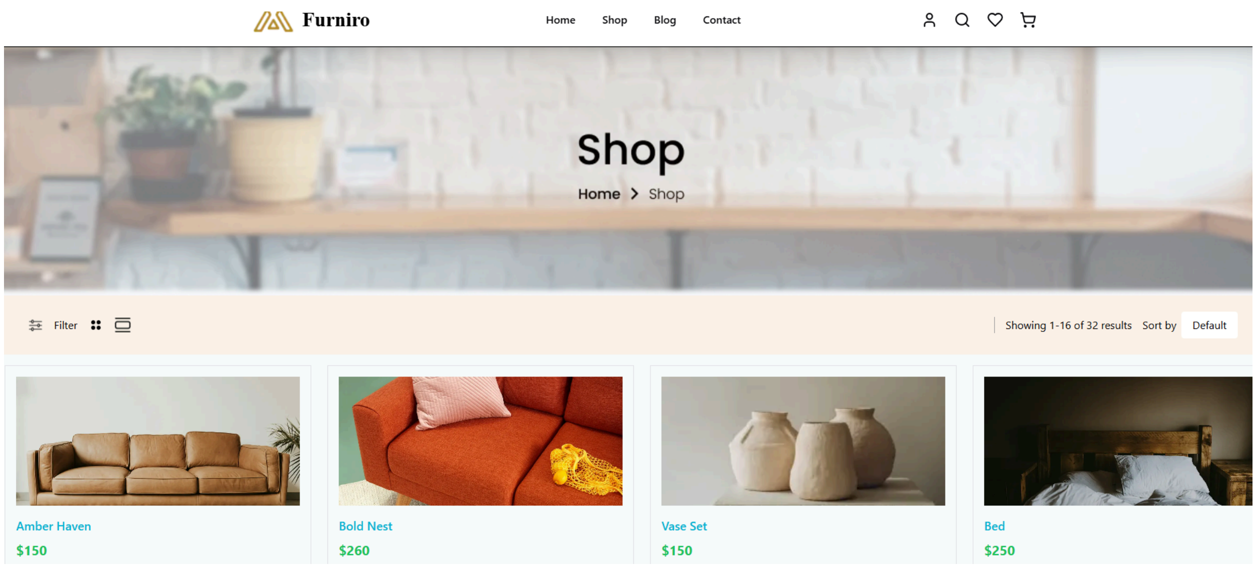
Laviosa
Rp 2.500.000
Out of Stock

Loliota
Rp 7.000.000
In Stock

Loliota
Rp 7.000.000
In Stock

2) Product Detail Component:

Then we move towards the Product detail components Where we make dynamic routes with the help of using groq query make function asynchronous , using map function to manage multiple images and use client function which is responsible to make relation between nextjs and sanity cms we use client.fetch to fetch data from given api from sanity cms



Amber Haven
$150

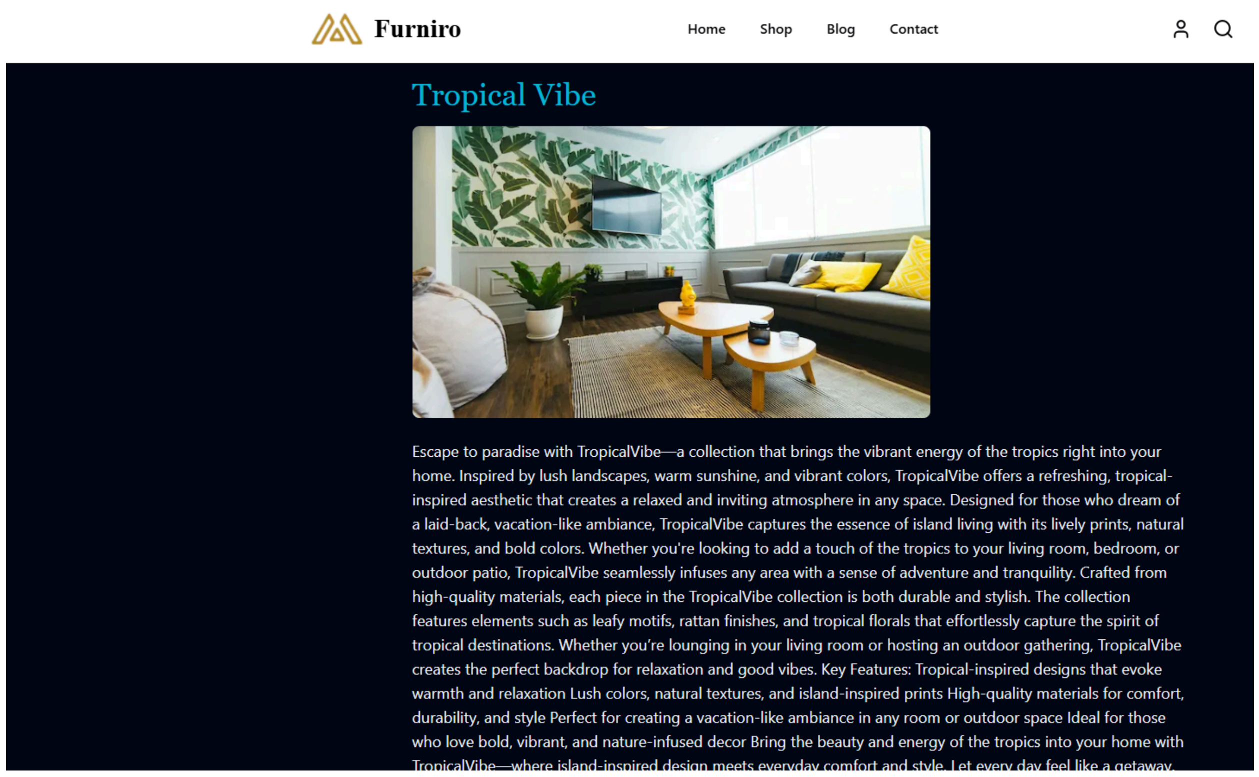
Bold Nest
$260

Vase Set
$150

Bed
$250

3) Dynamic Frontend:

This is my api data which i render with the help of sanity schema and fetch data from sanity cms with the help of groq query to make my ui more attractive and responsive we remove our data and show all the data which is present inside our api firstly import on sanity studio and then fetch data from sanity cms now we are Moving toward dynamic routing to make my api data dynamic for its description we use params and id to make connectivity between them we make two functions for product showing if my image cant show indicate product not found and if my data work properly then it shows images and description except of product not found we use tailwindcss to make and give my dynamic data to attractive ui.

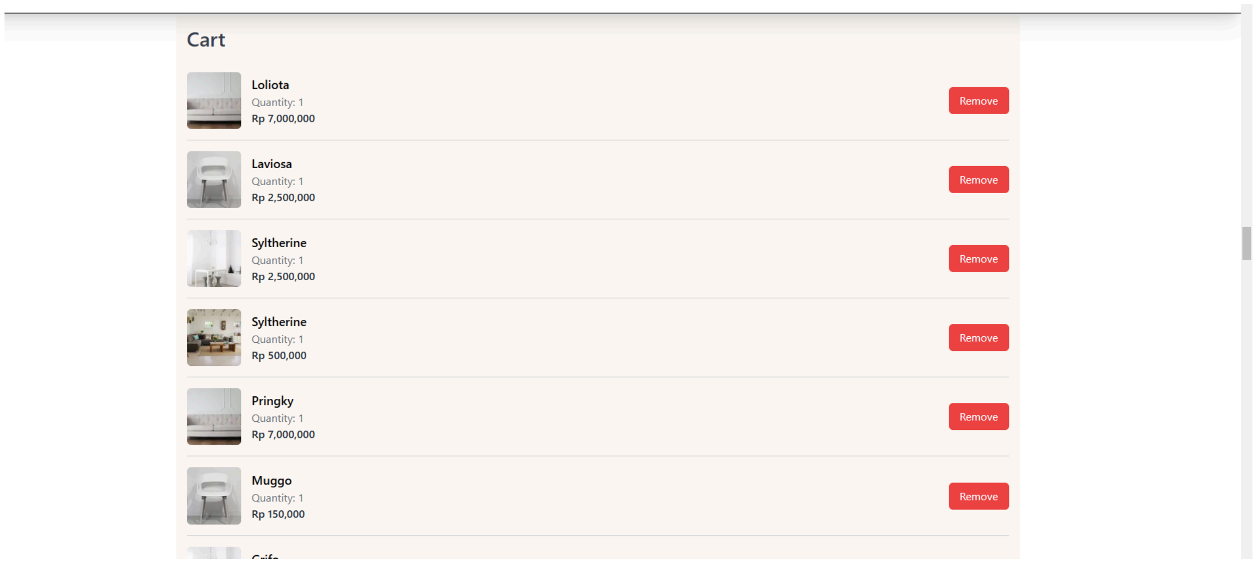


Tropical Vibe

Escape to paradise with TropicalVibe—a collection that brings the vibrant energy of the tropics right into your home. Inspired by lush landscapes, warm sunshine, and vibrant colors, TropicalVibe offers a refreshing, tropical-inspired aesthetic that creates a relaxed and inviting atmosphere in any space. Designed for those who dream of a laid-back, vacation-like ambiance, TropicalVibe captures the essence of island living with its lively prints, natural textures, and bold colors. Whether you're looking to add a touch of the tropics to your living room, bedroom, or outdoor patio, TropicalVibe seamlessly infuses any area with a sense of adventure and tranquility. Crafted from high-quality materials, each piece in the TropicalVibe collection is both durable and stylish. The collection features elements such as leafy motifs, rattan finishes, and tropical florals that effortlessly capture the spirit of tropical destinations. Whether you're lounging in your living room or hosting an outdoor gathering, TropicalVibe creates the perfect backdrop for relaxation and good vibes. Key Features: Tropical-inspired designs that evoke warmth and relaxation Lush colors, natural textures, and island-inspired prints High-quality materials for comfort, durability, and style Perfect for creating a vacation-like ambiance in any room or outdoor space Ideal for those who love bold, vibrant, and nature-infused decor Bring the beauty and energy of the tropics into your home with TropicalVibe—where island-inspired design meets everyday comfort and style. Let every day feel like a getaway.

4) <u>Cart Components:</u>

We make Cart component with the help of shadcnui tailwindcss and nextjs in our cart components you can easily add and remove data from cart and they give you message as well for the addition and confirmation of your product we use useState to make my Cart Component Functional.



Cart

Loliota
Quantity: 1
Rp 7,000,000

Laviosa
Quantity: 1
Rp 2,500,000

Syltherine
Quantity: 1
Rp 2,500,000

Syltherine
Quantity: 1
Rp 500,000

Pringky
Quantity: 1
Rp 7,000,000

Muggo
Quantity: 1
Rp 150,000

In React, `useState` is a built-in **hook** that allows you to add and manage state in functional components. State is essentially data that a component can hold, monitor, and update dynamically during its lifecycle.

How `useState` Works

- **Initial State**: You provide an initial value (e.g., `0`, `""`, `[]`, or an object).

- **State Value**: It returns a variable that holds the current state value.
- **State Setter**: It also returns a function to update the state.

Syntax

javascript

CopyEdit

```
const [state, setState] =
useState(initialValue);
```

- `state`: The current state value.
- `setState`: A function to update the state.
- `initialValue`: The initial value of the state.

What is `useState` in React?

In React, **`useState`** is a built-in **hook** that allows you to add and manage state in functional components. State is essentially data that a component can hold, monitor, and update dynamically during its lifecycle.

How `useState` Works

- **Initial State**: You provide an initial value (e.g., `0`, `""`, `[]`, or an object).
- **State Value**: It returns a variable that holds the current state value.
- **State Setter**: It also returns a function to update the state.

Syntax

javascript
CopyEdit

```
const [state, setState] =
useState(initialValue);
```

- `state`: The current state value.
- `setState`: A function to update the state.
- `initialValue`: The initial value of the state.

Example: Counter

javascript
CopyEdit

```
import React, { useState } from
"react";
```

```
function Counter() {
  // Declare a state variable "count"
with an initial value of 0
  const [count, setCount] =
useState(0);

  return (
    <div>
      <p>Current Count: {count}</p>
      <button onClick={() =>
setCount(count + 1)}>Increase</button>
      <button onClick={() =>
setCount(count - 1)}>Decrease</button>
    </div>
  );
}

export default Counter;
```

Key Points

1. **Initial Value**: The `useState` hook initializes the state with the provided value (e.g., `0`).

2. **Updating State**: Use the `setState` function to update the state. Calling `setState` triggers a re-render of the component.
3. **Immutability**: React state is immutable, meaning the state value should not be modified directly (e.g., never use `state = newValue`).

Common Use Cases

Form Inputs
Track user input values dynamically:
javascript
CopyEdit

```
const [name, setName] = useState("");
```

1.

Toggle Feature
Handle UI toggling (e.g., show/hide):
javascript
CopyEdit

```
const [isVisible, setIsVisible] =
useState(false);
```

2.

API Data
Store data fetched from an API:

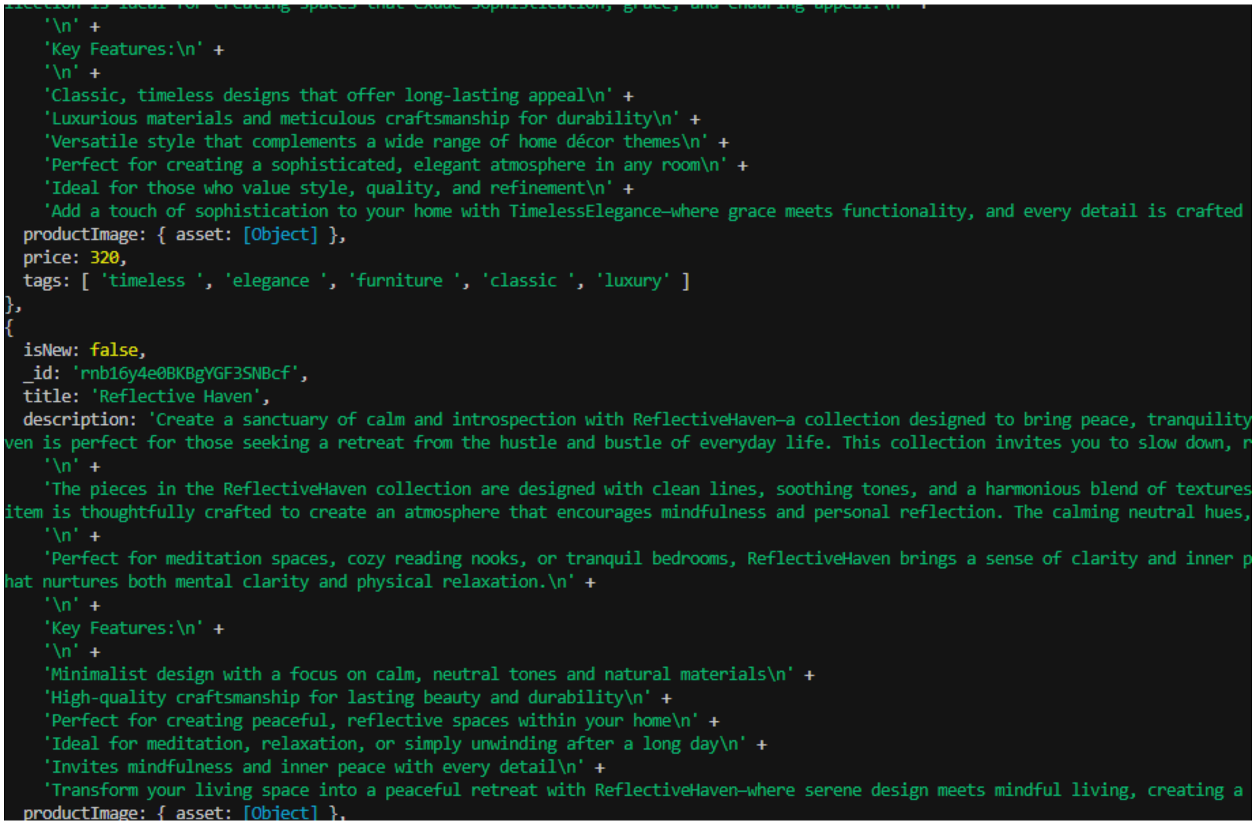
javascript
CopyEdit
```
const [products, setProducts] =
useState([]);
```

3.

Advantages of using use state in cart components:

- Easy to use and understand.
- Makes functional components stateful without converting them to class components.
- Works seamlessly with React's reactivity system.

Data importing in our Terminal is completely store in our dataset which is perfectly store in my Sanity cms you just see my imported data from api in terminal for the confirmation of accurate data transfer.



```
llection is ideal for creating spaces that exude sophistication, grace, and enduring appeal.\n' +
    '\n' +
    'Key Features:\n' +
    '\n' +
    'Classic, timeless designs that offer long-lasting appeal\n' +
    'Luxurious materials and meticulous craftsmanship for durability\n' +
    'Versatile style that complements a wide range of home décor themes\n' +
    'Perfect for creating a sophisticated, elegant atmosphere in any room\n' +
    'Ideal for those who value style, quality, and refinement\n' +
    'Add a touch of sophistication to your home with TimelessElegance—where grace meets functionality, and every detail is crafted
  productImage: { asset: [Object] },
  price: 320,
  tags: [ 'timeless ', 'elegance ', 'furniture ', 'classic ', 'luxury' ]
},
{
  isNew: false,
  _id: 'rnb16y4e0BKBgYGF3SNBcf',
  title: 'Reflective Haven',
  description: 'Create a sanctuary of calm and introspection with ReflectiveHaven—a collection designed to bring peace, tranquility
ven is perfect for those seeking a retreat from the hustle and bustle of everyday life. This collection invites you to slow down, r
    '\n' +
    'The pieces in the ReflectiveHaven collection are designed with clean lines, soothing tones, and a harmonious blend of textures
item is thoughtfully crafted to create an atmosphere that encourages mindfulness and personal reflection. The calming neutral hues,
    '\n' +
    'Perfect for meditation spaces, cozy reading nooks, or tranquil bedrooms, ReflectiveHaven brings a sense of clarity and inner p
hat nurtures both mental clarity and physical relaxation.\n' +
    '\n' +
    'Key Features:\n' +
    '\n' +
    'Minimalist design with a focus on calm, neutral tones and natural materials\n' +
    'High-quality craftsmanship for lasting beauty and durability\n' +
    'Perfect for creating peaceful, reflective spaces within your home\n' +
    'Ideal for meditation, relaxation, or simply unwinding after a long day\n' +
    'Invites mindfulness and inner peace with every detail\n' +
    'Transform your living space into a peaceful retreat with ReflectiveHaven—where serene design meets mindful living, creating a
  productImage: { asset: [Object] },
```

5) Reviews and Ratings Component:

For making Review and rating components we use multiple categories of function where user can easily drop their thoughts and share their feedback as per depend and choice of our products and we value their ratings and reviews we use:

`handleInputChange` is a common event handler function used in React to manage changes in form input fields (e.g., text boxes, checkboxes). It captures the value entered by the user and updates the corresponding state.

`handleSubmitReview` is a function typically used in a form submission context to handle the process of submitting a review (or any other data). It processes the review data and sends it to a server or updates the UI.

`preventDefault` is a method provided by the JavaScript `Event` object. It stops the browser's default action for an event.

`HTMLInputElement` is a built-in JavaScript interface representing `<input>` elements in the DOM. It provides properties and methods for interacting with input fields (e.g., `value`, `checked`, `focus()`).

`HTMLTextAreaElement` is a built-in JavaScript interface representing `<textarea>` elements in the DOM. It includes properties like `value`, `rows`, and `cols` to manage text input areas.

Summary of Concepts

Term	Definition

`handleInputChange`	A React function to handle changes in input fields and update the state.
`handleSubmitReview`	A React function to handle form submissions for submitting reviews or other data.
`preventDefault`	A method to stop the browser's default behavior for an event.
`HTMLInputElement`	Represents `<input>` elements in the DOM, used to interact with input fields.
`HTMLTextAreaElement`	Represents `<textarea>` elements in the DOM, used to interact with multi-line text input areas.

6)Footer and Header Components:

Objective

We Design and develop consistent, reusable components for navigation and branding to enhance user experience and accessibility.

Key Features to Implement

1. **Header Component**

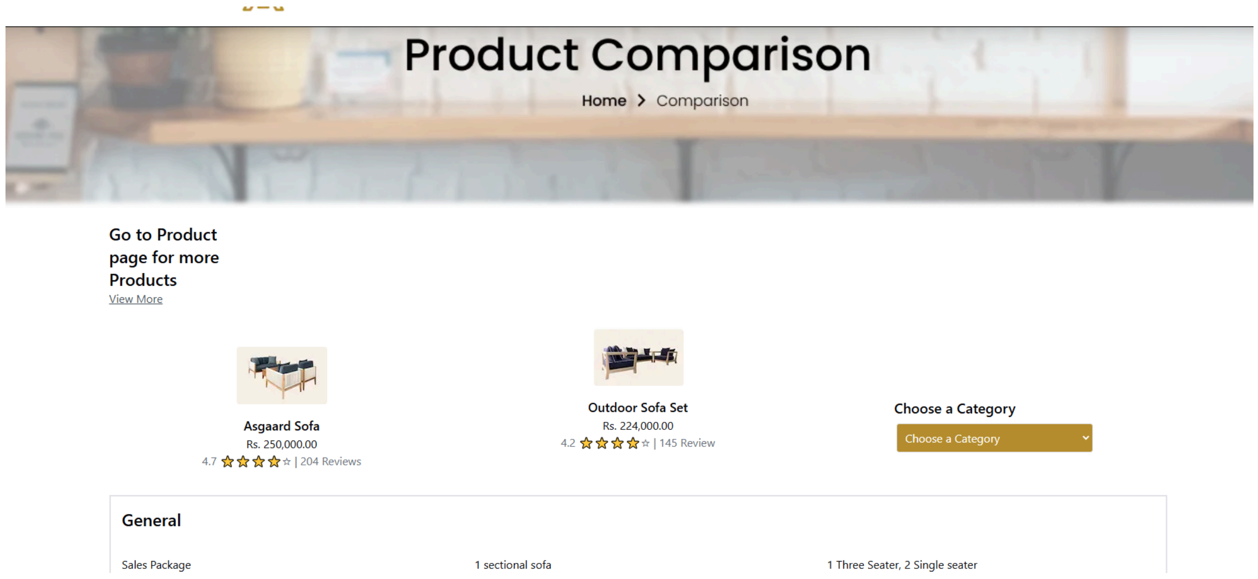
    - We Include the site's logo or branding.
    - Add navigation links to key pages, such as:
        - Home
        - About
        - Contact
        - Other important sections .
    - Implement a responsive menu which is completely mobile responsive.

2. **Footer Component**

    - We Add quick links to key pages (similar to the header).
    - Display social media icons with links.
    - Provide a contact section (e.g., email address, phone number, place order section, Thankyou).
        - We make a mobile-friendly and responsive header and footer.

Responsiveness: We make Ensure the components work well on all screen sizes, from desktop to mobile.

7) Product Comparison Component:



Go to Product page for more Products
View More

Asgaard Sofa
Rs. 250,000.00
4.7 ★★★★☆ | 204 Reviews

Outdoor Sofa Set
Rs. 224,000.00
4.2 ★★★★☆ | 145 Review

Choose a Category
Choose a Category

General

Sales Package	1 sectional sofa	1 Three Seater, 2 Single seater

Our website provide user product Comparision and side by side customization which make a competitive sense to users and in which users leads with multiple steps including

productSizing

Price

Tags

Description

Features etc

Summary: Dynamic Frontend Components for Marketplace

Objective
Focus on creating modular, reusable, and responsive frontend components to display marketplace data fetched from Sanity CMS or APIs, emphasizing professional workflows.

Key Components

1. **Product Listing Component**
   - Display product details in a grid layout.
   - Fields: Product Name, Price, Image, Tags.
2. **Product Detail Component**
   - Dynamic routing with GROQ queries and asynchronous functions.
   - Use `client.fetch` for data retrieval and `map()` for handling multiple images.
   - TailwindCSS ensures responsive UI.
3. **Cart Component**
   - Built with TailwindCSS, ShadCN UI, and Next.js.
   - Use `useState` for managing cart state (add/remove items, confirmations).
4. **Reviews and Ratings Component**
   - Features for users to leave feedback and view aggregated ratings.

    - React functions like `handleInputChange` and `handleSubmitReview` manage user input and submission.
    - Leverage `preventDefault`, `HTMLInputElement`, and `HTMLTextAreaElement` for form handling.
5. **Footer and Header Components**
    - **Header**: Logo, navigation links (e.g., Home, About, Contact), and a responsive mobile menu.
    - **Footer**: Quick links, social media icons, contact details, and a mobile-friendly design.
6. **Product Comparison Component**
    - Enables users to compare products side-by-side.
    - Features include size, price, tags, description, and unique features.

Additional Highlights

- **Dynamic Frontend**: Data rendered dynamically using GROQ queries, ensuring responsive and engaging UI.
- **State Management**: React's `useState` hook simplifies state handling in components like the cart and review forms.

- **Responsiveness**: All components are optimized for desktop and mobile views.

Dynamic Frontend Components Development Checklist

General Setup

- ✅ Ensure the project is connected to **Sanity CMS** or an API.
- ✅ Test data fetching with **GROQ queries** and validate data availability in the terminal.
- ✅ Use **TailwindCSS** for styling across all components.

Key Components

1. **Product Listing Component**
   - ✅ Render product data in a **grid layout**.
   - ✅ Include the following fields:
     - ✅ Product Name
     - ✅ Price
     - ✅ Image
     - ✅ Tags
   - ✅ Ensure layout is responsive across devices.
2. **Product Detail Component**
   - ✅ Implement **dynamic routing** with Next.js.

   - ✅ Fetch data using `client.fetch()` from Sanity CMS.
   - ✅ Use `map()` to display multiple product images.
   - ✅ Handle errors for missing data (e.g., "Product not found").
   - ✅ Design a visually appealing, responsive layout for product descriptions.
3. **Cart Component**
   - ✅ Add functionality to add/remove items from the cart.
   - ✅ Display confirmation messages for user actions.
   - ✅ Use `useState` for state management.
   - ✅ Ensure cart totals update dynamically.
   - ✅ Design for desktop and mobile responsiveness.
4. **Reviews and Ratings Component**
   - ✅ Create a form for users to submit reviews and ratings.
   - ✅ Display aggregated ratings and individual reviews.
   - ✅ Use functions like `handleInputChange` and `handleSubmitReview` for form handling.
   - ✅ Ensure accessibility and proper validation for form inputs.

5. **Footer and Header Components**
    - ✅ Build a **Header** with:
        - ✅ Logo or branding.
        - ✅ Navigation links (e.g., Home, About, Contact).
        - ✅ Responsive mobile menu.
    - ✅ Build a **Footer** with:
        - ✅ Quick links to important pages.
        - ✅ Social media icons with links.
        - ✅ Contact details (e.g., email, phone, address).
    - ✅ Ensure both components are mobile-friendly and responsive.
6. **Product Comparison Component**
    - ✅ Allow users to compare products side-by-side.
    - ✅ Include features like:
        - ✅ Product Size
        - ✅ Price
        - ✅ Tags
        - ✅ Description
        - ✅ Unique Features